

Session 3: Network Discovery and Mapping

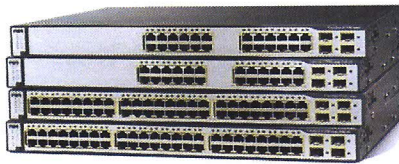
During this session, we will look at network and discovery mapping to help you begin to learn how to protect and defend your networks against attack.

LO4: Explain basic networking concepts necessary for active and passive network discovery

There are basic networking discovery concepts necessary for doing active and passive network discovery.

Basic Networking

- Why do devices need to communicate?
- When do devices need to communicate?
- PLC, RTU, IED, I/O Server, FEP, etc.
 - Transition from serial to Ethernet-based communication
- Common Ethernet components:
 - Network Interface Card (NIC)
 - Hub/repeater
 - Switch
 - Router
 - Firewall.



Basic Networking Components

Network interface card (NIC) — OSI Layer 1 & 2 device

- OSI Unique Layer 2 address (MAC)
- Promiscuous mode passes all traffic to the kernel.

Switched (Switch) — Typically OSI Layer 2

- Packets typically delivered to the intended port
- Reduced or no collision domain.

Non-switched (Hub) — OSI Layer 1

- Packets are delivered to every port
- Shared collision domain.

Router — Typically OSI Layer 3

- Directs traffic by comparing the destination address to routing table entries and passes it to the next hop.

Learning Objective 4

The first 6 characters of the MAC address are the OUI (organizationally unique identifier) of the network card.

Refer to <http://standards.ieee.org/develop/regauth/oui/oui.txt> for the list of vendors and associated OUIs.

Activity: Run the Windows “tracert” or the Unix “traceroute” command sometime to a destination, such as www.google.com and note how many networks or hops it traverses to reach the destination.

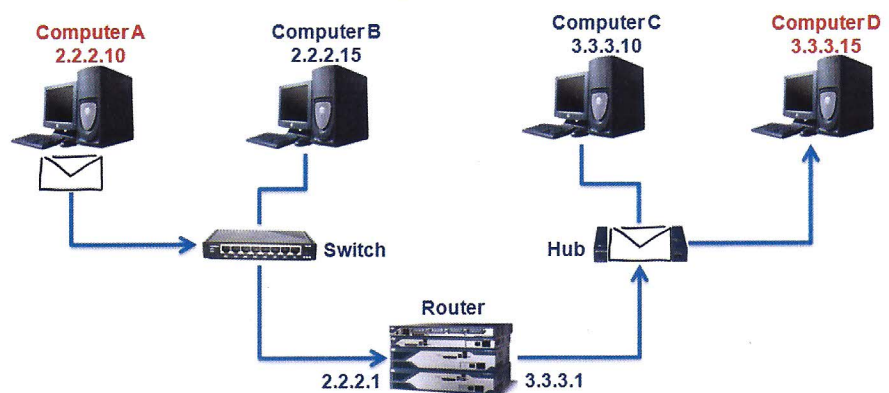
The Internet is governed by a set of documents referred to as **Request For Comments (RFC)**. A RFC is a publication of the Internet Engineering Task Force (IETF) and the Internet Society, the principal technical development and standards-setting bodies for the Internet. Check out <https://www.ietf.org/rfc.html>

Firewall — Allow or deny traffic based on source and destination information

- Only allow required traffic (general rule of thumb)
- Packet Filtering: OSI Layers 1 – 4
 - Stateless: No session information stored, faster and simpler
 - Stateful: Store session information for active sessions, slower and more complex.
- Application: Understands application layer protocols like HTTP, FTP, DNS, etc.

Basic Packet Traversal

There are basic packet traversal concepts and ways a packet can traverse several networks to reach a specified destination. The various routes traversed by a packet to reach the designated destination are referred to as hops.



Network Address Translation (NAT)

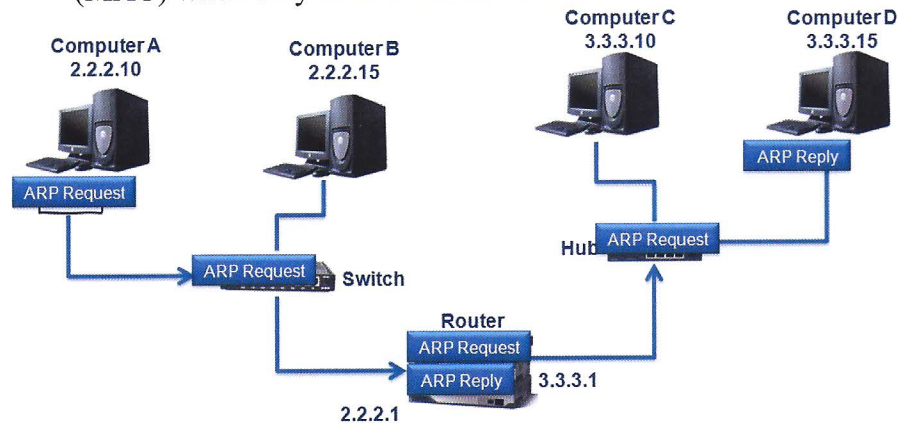
- Public address assigned to computer or group of computers inside a private network
- Limits number of IP address; economical and increases security.

Internet Assigned Numbers Authority (IANA) private addresses

- Nonroutable – cannot be transmitted onto public Internet
- 10.0.0.0 – 10.255.255.255
- 172.16.0.0 – 172.31.255.255
- 192.168.0.0 – 192.168.255.255.

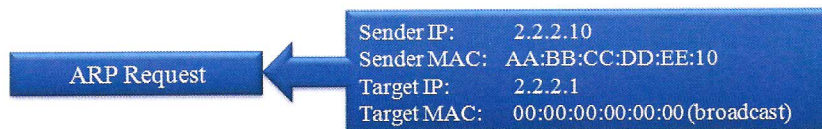
Address Resolution Protocol (ARP)

- Protocol for determining a network host's hardware address (MAC) when only its IP address is known

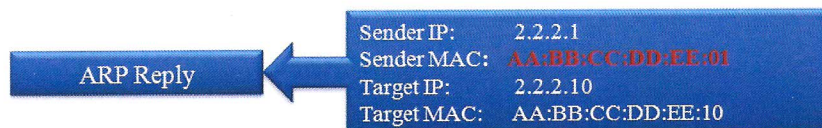


Address Resolutions Protocol (ARP) packet contents

- Computer A — ARP Request



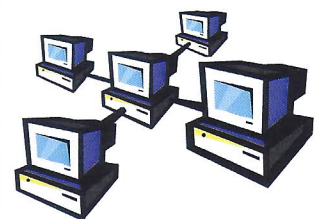
- Router — ARP Reply



The MAC address in red is the address that was “requested” by the requestor in order to send the data destined for the target IP in the original ARP request.

Transmission control protocol (TCP)

- Reliable stateful communication
- 3-way handshake
 - SYN:** The active open is performed by the client sending a SYN to the server. It sets the segment's sequence number to a random value A.
 - SYN-ACK:** In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number ($A + 1$), and the sequence number that the server chooses for the packet is another random number, B.



- **ACK:** Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgment value i.e., $A + 1$, and the acknowledgment number is set to one more than the received sequence number, i.e., $B + 1$.

TCP is:

- **Reliable** – It manages message acknowledgment, retransmission, and timeout. Multiple attempts to deliver the message are made. If a message gets lost along the way, the server will re-request the lost part.
- **Ordered** – If two messages are sent over a connection in sequence, the first message will reach the receiving application first. When data segments arrive in the wrong order, TCP buffers the out-of-order data until all data can be properly reordered and delivered to the application.
- **Heavyweight** – It requires three packets to set up a socket connection before any user data can be sent. TCP handles reliability and congestion control.
- **Streaming** – Data are read as a byte stream, no distinguishing indications are transmitted to signal message (segment) boundaries.

User datagram protocol (UDP)

- Unreliable stateless communication
- No handshaking.

VoIP

UDP is:

- **Unreliable** – When a message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, retransmission, or timeout.
- **Not ordered** – If two messages are sent to the same recipient, the order in which they arrive cannot be predicted.
- **Lightweight** – There is no ordering of messages, tracking connections, etc. It is a small transport layer designed on top of IP.
- **Datagrams** – Packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries that are honored upon receipt, meaning a read operation at the receiver socket will yield an entire message as it was originally sent.
- **No congestion control** – UDP itself does not avoid congestion, and it's possible for high bandwidth applications to trigger congestion collapse, unless they implement congestion control measures at the application level.

Internet Control Message Protocol (ICMP)

- Provides control, troubleshooting, and error messages
- Used by ping and traceroute commands.

(ping)

An ICMP packet can be one of various types. ICMP is:

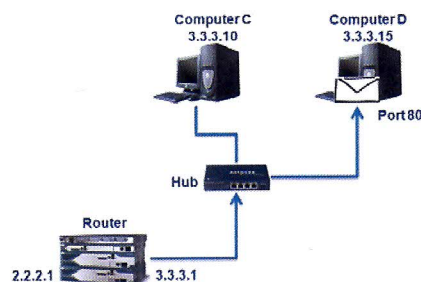
- Used to announce network errors such as a host or entire network being unreachable.
- Used to announce network congestion (congestion control) by informing packet senders to reduce the send rate using the "Source Quench" ICMP type message.
- Used to assist in troubleshooting. The "ping" command uses an ICMP "Echo Request" message. The reply is an ICMP "Echo Reply" message.
- Used to announce timeouts. The traceroute command relies on ICMP "Time Exceeded" packets returned from various routers to detail the route from a source to a destination.

TCP/UDP

- Well-known 0 – 1023 (root to bind)
- Registered 1024 – 49151
- Dynamic/private 49152 – 65535.

Examples of ICS services

- E/IP – Allen Bradley which stems from CIP protocol (44818)
- ICCP (Inter Control Center Protocol) – Typically used in the energy sector for passing load balancing information (102).
- Modbus – Multi-vendor use of this protocol used to communicate with PLC, RTUs, etc. (502).
- OPC – OLE (Object Linking and Embedding) for Process Control
- DNP3 – Distributed Network Protocol typically used in utilities such as electricity and water companies. Used to communicate with PLCs, RTUs, etc. (20000).
- PROFINET – Industrial Ethernet standard; uses TCP/IP and is, in effect, a real-time Ethernet.



*Dynamic is sometimes referred to as an **ephemeral port**. Port allocations are temporary and only valid for the duration of the communication session.*



Learning Objective 5

LO5: Discuss passive discovery

There are two types of network discovery: passive and active. First, we will focus on passive discovery.

Passive Discovery

What is **passive** network discovery?

- Using information stored locally on a compromised host to identify new host and network targets
- Attempt to identify new targets without sending any network packets.

Why perform **passive** network discovery?

- More difficult to detect than active discovery
- May provide valuable information that active discovery cannot
- Active discovery not always possible, i.e., ICS systems.

There are many tools and system commands to aid in the network discovery process.

Much can be learned from computer configuration files such as services being used, hosts that may be accessed on a frequent basis, hosts designated as domain name servers, etc.

History files provide information as to what the host is being used, commands issued, processes running, etc.

Caches provide information that various system processes store as information is received and processed. Commands to retrieve cached data and researching files that are cached provide important information on users, networks accessed, etc.

Tools	History Files
Tcpdump, Wireshark Ipconfig (windows) Ifconfig (linux) Netstat Arp Net Route Iptables EtherApe (GUI)	.bash_history RDP Log Files
Configuration Files	Caches
Custom Scripts (cron, start-up) Apache (mysql, etc.) Resolv.conf, hosts	Arp Nbtstat DNS Browser

Ipconfig

The windows ipconfig command displays the IP address, subnet mask, and default gateway for all adapters. The /all parameter displays the full TCP/IP configuration.

- Command: **ipconfig /all**

```
C:\WINDOWS\system32\cmd.exe
C:\>ipconfig /all

Windows IP Configuration

    Host Name . . . . . : idaho-b2a8de164
    Primary Dns Suffix . . . . . : 
    Node Type . . . . . : Unknown
    IP Routing Enabled. . . . . : No
    WINS Proxy Enabled. . . . . : No
    DNS Suffix Search List. . . . . : localdomain

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix . : localdomain
    Description . . . . . : VMware Accelerated AMD PCNet Adapter
    Physical Address. . . . . : 00-0C-29-1E-24-96
    Dhcp Enabled. . . . . : Yes
    Autoconfiguration Enabled . . . . : Yes
    IP Address. . . . . : 192.168.90.128
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.90.2
    DHCP Server . . . . . : 192.168.90.254
    DNS Servers . . . . . : 192.168.90.2
    Lease Obtained. . . . . : Wednesday, May 19, 2010 11:23:22 AM
    Lease Expires . . . . . : Wednesday, May 19, 2010 11:53:22 AM

C:\>
```

Ifconfig

Ifconfig (short for interface configuration) is a system administration utility in Unix-like operating systems to configure, control, and query TCP/IP network interface parameters.

- Command: **/sbin/ifconfig -a**

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# /sbin/ifconfig -a
eth0
    Link encap:Ethernet  HWaddr 00:0c:29:e3:10:e6
    inet addr:172.16.255.141 Bcast:172.16.255.255 Mask:255.255.255.0
    inet6 addr: fe80::20c:29ff:fee3:10e6/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
    RX packets:26416 errors:0 dropped:0 overruns:0 frame:0
    TX packets:15037 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:29605461 (28.2 MiB)  TX bytes:1560405 (1.4 MiB)
    Interrupt:19 Base address:0x2000

lo
    Link encap:Local Loopback
    inet addr:127.0.0.1 Mask:255.0.0.0
    inet6 addr: ::1/128 Scope:Host
    UP LOOPBACK RUNNING  MTU:65536  Metric:1
    RX packets:1525 errors:0 dropped:0 overruns:0 frame:0
    TX packets:1525 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:257391 (251.3 KiB)  TX bytes:257391 (251.3 KiB)

root@kali:~/Desktop#
```

The two interfaces shown are the wired NIC (eth0) and the localhost interface (lo). Depending on the configuration of your system, you may see more interfaces in the list. For example:

- eth1, eth2,...if you have more than 1 one wired NIC
- vmnet1 and vmnet8 used by the VMware Host OS
- wlan0 for a wireless interface.

The most useful information displayed by ifconfig includes:

HWaddr – Interface MAC address

inet addr – Interface IP address

Bcast – Network broadcast address

Mask – Network mask (akanetmask)

inet6 addr – Ipv6 address.

Common uses for ifconfig include setting an interface's IP address, netmask, and disabling or enabling a given interface. Many UNIX-like operating systems initialize their network interfaces with shell-scripts that call ifconfig. As an interactive tool, system administrators routinely use the utility to display and analyze network interface parameters.

Address Resolution Protocol (ARP)

What is ARP?

- Address resolution protocol (ARP) is used to find the media access control (MAC) address of a network neighbor for a given IPv4 address
- Tool used to view the ARP table of the local machine.
- ARP is also used to forward IP datagrams to local routers for destinations that are not on the local network.

Why look at the ARP table?

- Lists all the hosts with which the host has recently communicated.

Note: The `-n` option disables the DNS lookup to resolve the host name. If you do not use it, the `arp` command becomes an **active** tool instead of a **passive** tool!

Command Usage:

- Windows Command: `arp -a`

```

C:\WINDOWS\system32\cmd.exe
C:\>arp -a

Interface: 192.168.90.128 --- 0x10003
Internet Address      Physical Address      Type
192.168.90.1          00-50-56-c0-00-08    dynamic
192.168.90.2          00-50-56-fd-08-00    dynamic
192.168.90.254        00-50-56-ff-22-17    dynamic
  
```

- Linux Command: `arp -a -n -i eth0`
`arp -n -i eth0`

```

root@kali: ~/Desktop
File Edit View Search Terminal Help

root@kali:~/Desktop# arp -a -n -i eth0
? (172.16.255.254) at 00:50:56:e3:f7:ca [ether] on eth0
? (172.16.255.2) at 00:50:56:ea:43:5c [ether] on eth0

root@kali:~/Desktop# arp -n -i eth0
Address      Hwtype  Hwaddress      Flags Mask      Iface
172.16.255.254 ether    00:50:56:e3:f7:ca C                eth0
172.16.255.2  ether    00:50:56:ea:43:5c C                eth0
root@kali:~/Desktop#
  
```

→Note that the IP address and the MAC address for each host is displayed. Both the Windows and Linux commands display entries for each network interface card.

The interface option (`-i eth0`) is used to select only ARP cache entries for the specific interface. This should be used if you have a wireless interface AND a wired interface.

There are two output formats for the Linux `arp` command. The normal format is for System5. If you prefer the Berkley Standard Distribution (BSD), use the `-a` option.

ARP-scan sends ARP packets to hosts on the local network and displays any responses that are received. By default, the ARP

arp - scan

packets are sent to the Ethernet broadcast address, **ff:ff:ff:ff:ff:ff**.

Arp-scan supports Ethernet and 802.11 wireless networks. It can also support token ring and FDDI, but they have not been tested. The ARP protocol is an IPv4 layer-2 (datalink layer) protocol that is used to determine a host's layer-2 address given its layer-3 (network layer) address. ARP was designed to work with any layer-2 and layer-3 address format, but the most common use is to map IP addresses to Ethernet hardware addresses, and this is what arp-scan supports. ARP only operates on the local network, and cannot be routed. Although the ARP protocol makes use of IP addresses, it is not an IP-based protocol and ARP-scan can be used on an interface that is not configured for IP.

The target hosts to scan may be specified in one of three ways:

- Specifying the targets on the command line,
- Specifying a file containing the targets with the **--file** option,
- Specifying the **--localnet** option which causes all possible hosts on the network attached to the interface to be scanned.

For hosts specified on the command line, or with the **--file** option, you can use either IP addresses or hostnames. You can also use network specifications IPnetwork/bits, IPstart-IPend, or IPnetwork:NetMask.

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# arp-scan -g 192.168.10.0/24
Interface: eth0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.2 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
192.168.10.2 00:50:56:a0:48:fb VMware, Inc.
192.168.10.10 00:10:18:4e:2a:b0 BROADCOM CORPORATION
192.168.10.11 00:50:56:a0:2d:26 VMware, Inc.
192.168.10.12 00:50:56:a0:56:1b VMware, Inc.
192.168.10.21 00:50:56:a0:2f:e4 VMware, Inc.
192.168.10.22 00:50:56:a0:1b:d2 VMware, Inc.
192.168.10.32 00:50:56:a0:1c:b6 VMware, Inc.
192.168.10.40 00:a0:1d:30:b2:1c SIXNET
192.168.10.41 00:50:56:a0:22:b3 VMware, Inc.
192.168.10.42 00:50:56:a0:1d:a2 VMware, Inc.
192.168.10.50 00:50:56:a0:39:b3 VMware, Inc.
192.168.10.55 00:50:56:a0:4c:6d VMware, Inc.
192.168.10.66 00:50:56:a0:5f:f9 VMware, Inc.
192.168.10.97 00:50:56:a0:2d:b3 VMware, Inc.
192.168.10.99 54:42:49:7b:2c:10 Sony Corporation
192.168.10.254 00:19:e2:ab:32:8c Juniper Networks

102 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.2: 256 hosts scanned in 1.856 seconds (137.93 hosts/sec). 18 responded
root@kali:~/Desktop#
root@kali:~/Desktop#
```

Netstat

Netstat is a tool for looking at a host's active network connections, routing tables, and a number of network interface statistics.

It is available on Unix, Unix-like, and Windows NT-based operating systems.

ARP-scan:

<http://www.nta-monitor.com/tools-resources/security-tools/arp-scan>

*Devices
Juniper
VMware*



Why use netstat?

- Identify local services that could be used for privilege escalation
- Identify established connections (through firewall) to other networks

Netstat on a Windows system

- Windows Command: **netstat -nob**

```
C:\WINDOWS\system32\cmd.exe
C:\>netstat -nob

Active Connections

Proto Local Address          Foreign Address         State       PID
TCP   127.0.0.1:1033          127.0.0.1:40000        ESTABLISHED 164
[SFCTlCom.exe]
TCP   127.0.0.1:1059          127.0.0.1:40000        ESTABLISHED 164
[SFCTlCom.exe]
TCP   127.0.0.1:1060          127.0.0.1:40000        ESTABLISHED 164
[SFCTlCom.exe]
TCP   127.0.0.1:1061          127.0.0.1:40000        ESTABLISHED 164
[SFCTlCom.exe]
```

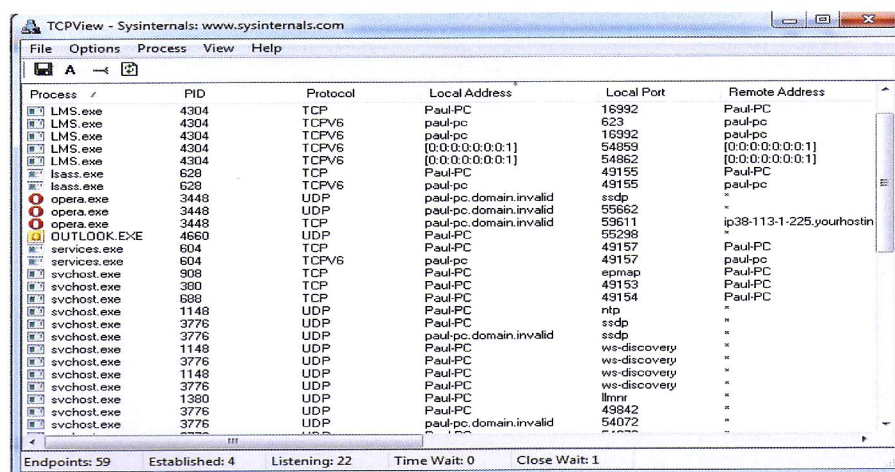
Netstat provides statistics for:

- Name of the protocol (TCP or UDP)
- IP address of the local computer and port number
- IP address and port number of remote computer
- State of a TCP connection
- Process name and ID of the connection.

The link for the tool set is:
<http://technet.microsoft.com/en-us/sysinternals/bb842062>

The Microsoft package, called “Security Essentials,” contains several advanced windows utilities that can help you to manage, troubleshoot, and diagnose Windows-based systems. **TCPView** shows all of the TCP and UDP connections including:

- Process
- Protocol
- Local IP address/Host name
- Remote IP address/Host name
- TCP state (ESTABLISHED, CLOSE_WAIT, and TIME_WAIT)



Netstat on a Linux system

- Linux Command: **netstat -pantu**

Reference:

<http://www.netstat.net/>

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# netstat -pantu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      5180/opensshd
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      5215/opensshd: wai
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      5229/opensshd: wai
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      5195/opensshd
tcp        0      0 172.16.255.141:58943    173.194.33.167:80      ESTABLISHED 5681/firefox
tcp        0      0 172.16.255.141:60551    173.194.33.163:80      ESTABLISHED 5681/firefox
tcp6       0      0 :::80                   :::*                    LISTEN      4448/apache2
udp        0      0 0.0.0.0:514             0.0.0.0:*               *          2062/rsyslogd
udp        0      0 0.0.0.0:58              0.0.0.0:*               *          3078/dhclient
udp        0      0 0.0.0.0:4373            0.0.0.0:*               *          3078/dhclient
udp6       0      0 :::24942                :::*                    *          3078/dhclient
udp6       0      0 :::514                  :::*                    *          2062/rsyslogd
root@kali:~/Desktop#
```

The output results of the Linux netstat command are similar to the Windows results. There are many different states defined for the connection results. The state definitions are detailed in RFC 793.*

Below details the options that are typically used with the netstat command.

Windows Command: **netstat -anob**

- a all sockets
- n no name resolution
- b owning process name
- o owning process ID

Linux Command: **netstat -pantu**

- p owning process ID
- a all sockets
- n no name resolution
- t tcp
- u udp

Browser History

An important aspect of passive discovery is reviewing the browser history. This allows you to identify web or other internal servers, user password, and auto complete information such as cache files, bookmarks, and favorites.

- Most browsers have a “clear” history and other obfuscation features making discovery more difficult
- However, tools are available to automate or bypass obfuscation techniques.

.bash_history file

The .bash_history file contains a history of the commands executed in the INTERACTIVE bash command shell. It is located in the

*There are over 7400 RFCs (request for comments) published by the Internet Engineering Task Force (IETF) and the Internet Society. RFC 793 is called “Transmission Control Protocol” and describes the standard transmission control protocol.

Note: the “-n” option specifies that host name resolution will not be attempted. If this option is omitted then there are considerable packets created and distributed by the name servers used by the host where the netstat command is run, making the use of the command considerably less passive.

user's home directory generally located at **/home/** for a normal user and **/root** for the root user. The **tilde** (~) can be used as shorthand for indicating the home directory. When used by itself, it means the home directory of the current logged in user. For example, **~/Desktop** refers to the user's Desktop directory. When used with a username, it means the home directory for the specified user. For example, **~root/Desktop** means root's Desktop directory.

You can also use the **history** command to view the **~/.bash_history** file.

Why look at the .bash_history file?

- May contain user passwords
- Identify host addresses specified with commands such as SSH, FTP, etc.
- Search history for command line tool usage such as SSH, FTP, telnet, etc.
- Command: **grep ssh ~/.bash_history**



```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# grep ssh ~/.bash_history
ssh 192.168.10.22
ssh 10.10.10.123
ssh 192.168.0.3
ssh 1.2.3.97
ssh 1.2.3.22
root@kali:~/Desktop# grep ssh ~root/.bash_history
ssh 192.168.10.22
ssh 10.10.10.123
ssh 192.168.0.3
ssh 1.2.3.97
ssh 1.2.3.22
root@kali:~/Desktop# history | grep ssh
139 grep ssh ~/.bash_history
140 grep ssh ~root/.bash_history
141 history | grep ssh
root@kali:~/Desktop#
```

The image shows the results of the **grep** (global regular expression print) command used against the **.bash_history** file. The command returned all the uses of **ssh** in the file and the host being accessed. Other commands to look for may include: **ftp**, **telnet**, **rlogin**, **rsh**, etc. It is also a good idea to look at the entire file. Browse it from top to bottom to see the sequence of commands that were executed. This may provide insight to other hosts that have been accessed, processes started or stopped, or scripts that have been used.

Routing Table

A routing table stores routes to particular network destinations and provides information about the topology of the network immediately around it.

*From Gateway
or
New Networks
gateway hosts*

Why look at the routing table?

- Identify router/gateway IP addresses
- Identify new network and host targets
- Gateway hosts great target for Man-in-the-Middle (MitM) attack
- Command: **route -n**

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          172.16.255.2   0.0.0.0         UG    0      0      0 eth0
172.16.255.0     0.0.0.0        255.255.255.0   U      0      0      0 eth0
root@kali:~/Desktop#
```

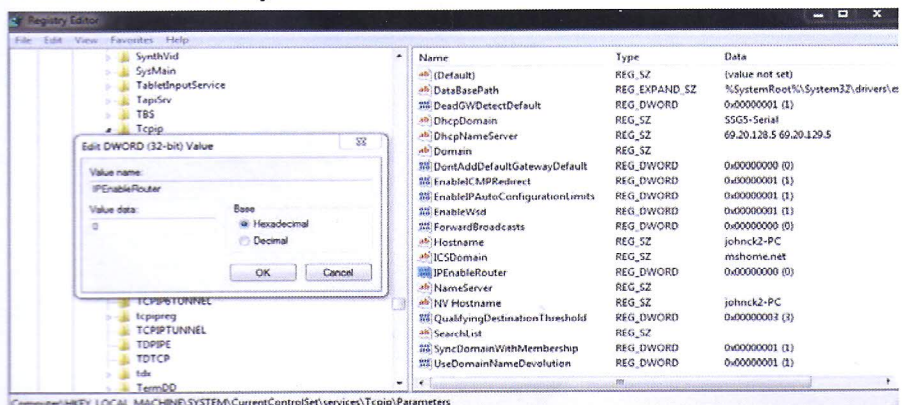
The above table tells us that there is one network interface card and gives us the default route. All traffic destined for the 192.168.90.0 and 169.254.0.0 subnets are directed to the default route as well.

How can you tell if this host is acting as a gateway?

- Not Forwarding = 0, Forwarding = 1
- Command: **head /proc/sys/net/ipv4/ip_forward**

```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# head /proc/sys/net/ipv4/ip_forward
0
root@kali:~/Desktop#
```

On a Windows based system, use **regedit** to search for the IPEnableRoute entry.



Possible Flags include:

- U** (route is up)
- H** (target is a host)
- G** (use gateway)
- R** (reinstate route for dynamic routing)
- D** (dynamically installed by daemon or redirect)
- M** (modified from routing daemon or redirect)
- A** (installed by addrconf)
- C** (cache entry)
- !** (reject route)

IP forwarding enables one workstation to sit on different networks and to act as gateway forwarding IP packets from one network to another. IP forwarding is also referred to as "bridging" networks. It requires at least two network cards installed with each card connected to a different network.

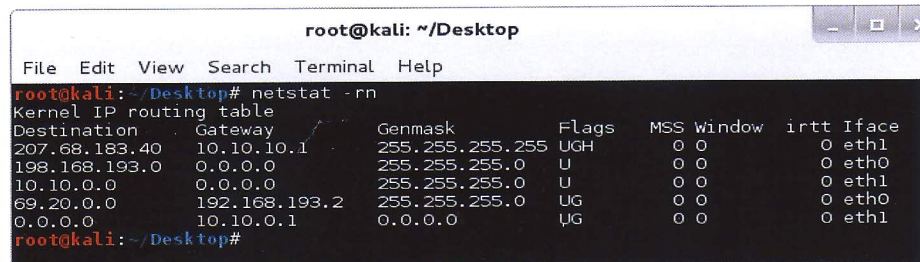
The image below shows how the various destination types are depicted in the routing table output. The route table is displayed using the netstat command parameters:

- **-r**: displays the routing table
- **-n**: no name resolution.

A host route specifies a particular host as the destination and has a subnet mask of 255.255.255.255.

The local and remote network destination indicates all IP addresses that fall within the subnet mask are to use the specified gateway.

The default route is the fallback destination route used by the router when no other known route exists for a given destination address.



```
root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali: ~/Desktop# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
207.68.183.40 10.10.10.1 255.255.255.255 UGH 0 0 0 eth1
198.168.193.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
10.10.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
69.20.0.0 192.168.193.2 255.255.255.0 UG 0 0 0 eth0
0.0.0.0 10.10.0.1 0.0.0.0 UG 0 0 0 eth1
root@kali: ~/Desktop#
```

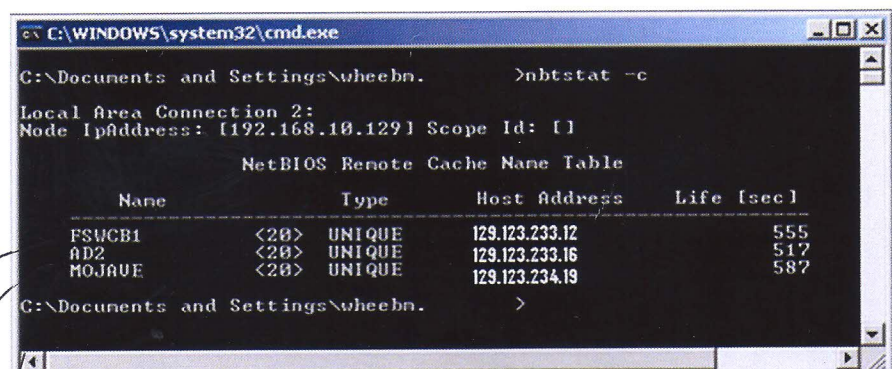
netBIOS

Network Basic Input/Output System (netBIOS) allows applications on different computers to communicate within a local area network. It is used by Microsoft File and Printer Sharing.

How can netBIOS be helpful?

- Discovers networks and hosts by looking at netBIOS cache
- Command: **nbtstat -c**

The nbtstat command depicted in the image shows the contents of the NetBIOS cache, the table of NetBIOS names and their resolved IP addresses of recently contacted systems.



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\wheebn. >nbtstat -c
Local Area Connection 2:
Node IpAddress: [192.168.10.129] Scope Id: []

NetBIOS Remote Cache Name Table

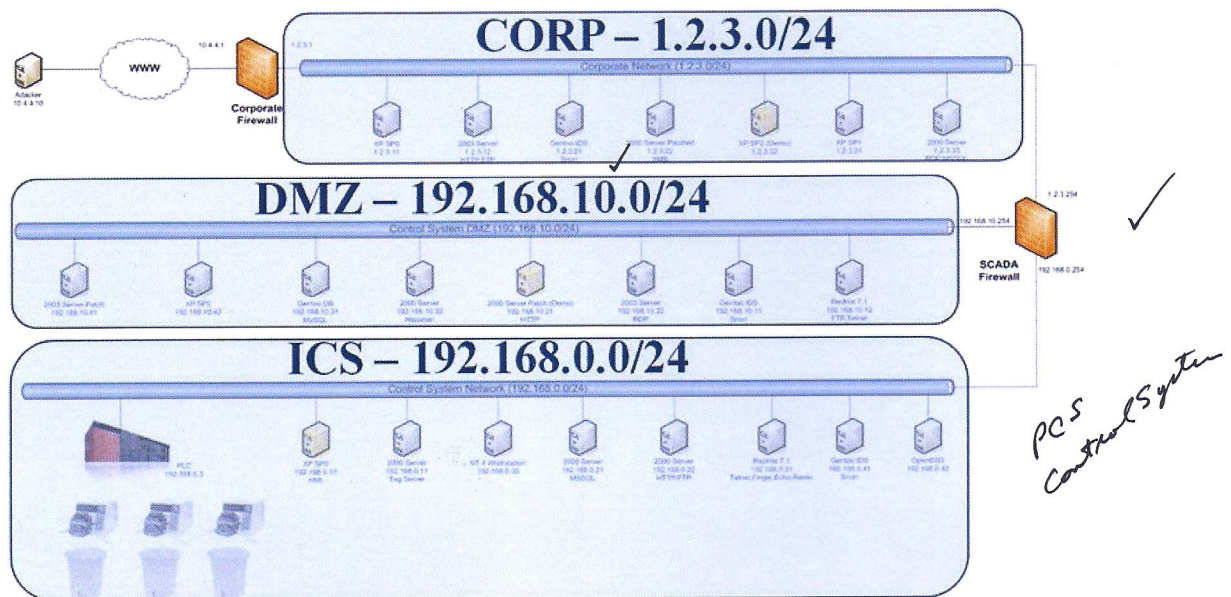
Name                Type                Host Address        Life [sec]
-----
FSUCB1              <20>                UNIQUE              129.123.233.12      555
AD2                  <20>                UNIQUE              129.123.233.16      517
MOJAVE              <20>                UNIQUE              129.123.234.19      587
C:\Documents and Settings\wheebn. >
```

File Shares
Active Directory
Print Servers

UNIX flavor boxes can run “nbtscan.” NBTscan scans IP networks for NetBIOS name information. It sends NetBIOS status query to each address in the supplied range and lists received information in human readable form. For each responding host, it lists the IP address, NetBIOS computer name, logged-in user name and MAC address. NetBIOS is only implemented by Windows (and some software on UNIX such as Samba). Consequently, nbtscan will only list Windows boxes and Linux boxes that have implemented Samba.

unix

Network Layout



This diagram shows a typical network configuration for a Control System. It shows the PLC connected to the control local area network (LAN). In some instances PLCs are directly connected to other networks to allow programming and configuration changes from remote workstations.

A DMZ is typical for most systems. The DMZ adds a level of isolation for the control LAN from the corporate network.

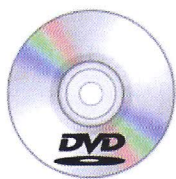
There are two Intrusion Detection points: one on the control LAN and one on the DMZ LAN.

Proper configuration of these devices is critical to the security of a networked domain, and especially important when connecting control systems to other networks.

IP Address

You will be provided an IP address for one of the following networks

- Corporate 1.2.3.0/24 – DHCP: 1.2.3.100 – 200
- DMZ 192.168.10.0/24 – DHCP: 192.168.10.100 – 200
- ICS 192.168.0.0/24 – DHCP: 192.168.0.100 – 200.



DVD provided to use for all hands-on exercises. You may also use your own tools.

The Desktop directory is located in the **root** user's home directory
/root, not the system root **/**.

Tools – Kali

Kali is a Linux-based penetration testing arsenal that aids security professionals in the ability to perform assessments in a purely native environment dedicated to hacking. The penetration distribution has been customized down to every package, kernel configuration, script and patch solely for the purpose of the penetration tester.

The version of Kali used in this class has been slightly modified to include files used during the exercises.

Because you are booting the laptop to a Live-DVD, any files you save to the Kali file structure will not be retrievable after you turn the computer off. Your internal hard drive may be accessible. However, if you can access the hard drive, there is a possibility that others in the classroom will also be able to access it. With this in mind, you are welcome to remove your hard drive from your laptop. Screw drivers are available if necessary.

All the hands-on exercises performed over the course of the training are done within the Kali-Live environment.

Start Kali:















1. Turn off and remove your hard drive (optional).
2. Insert and boot from the Kali DVD (may need to use F12 boot menu or change BIOS boot sequence).

Once the Kali boot menu appears press enter on the first option, "Kali Text – Default Boot Text Mode." If this option is not highlighted, use your arrow keys to move up or down until it is highlighted, then select it.

3. When the boot process is completed, the root user desktop should appear.



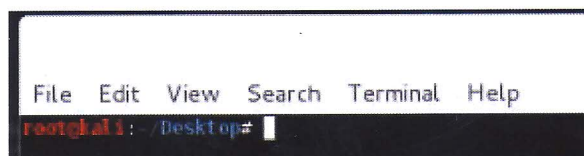
The icons found on the upper desktop panel and/or on the desktop have the following functions:

Icon	Function/Utility/Application	Icon	Function/Utility/Application
	Firefox Browser		Start OpenVAS Vulnerability Scanning Tool
	Stop OpenVAS		Iceweasel Browser (Firefox clone)
	Start a terminal Window aka Command Window		Edit files with Nedit
	Take screenshot Desktop, Cindow, Custom area		View image files Eye-of-Gnome
	Start Metasploit Framework + Postgresl		View PDF or PS files
	Zenmap Nmap Scanning Tool GUI		Wireshark Network Analysis
	Networkminer - Network Forensics		Install Kali to hard drive or VM-disk

4. Open a command shell by selecting the  icon on toolbar.



Looking at a close up of the terminal window, the text at the start of the line is the command prompt. This is issued to the terminal window after each command finishes.



The prompt has three parts:

root	- User name you used when you logged in
@kali	- Hostname of the system
~/Desktop	- Your current directory
#	- End of prompt

Note: your IP address will have the last octet between x.x.x.100 and x.x.x.199 (e.g., 1.2.3.100)

All commands in this class will be typed after the prompt. Pay close attention to which directory you are in; some exercises will require you to be in a specific directory.

5. Type **ifconfig** to identify your network interfaces
6. Type **dhclient eth0** to obtain an ip address

```

root@kali: ~/Desktop
File Edit View Search Terminal Help
root@kali:~/Desktop# ifconfig
eth0      Link encap:Ethernet  Hwaddr 00:1c:23:1b:d5:8f
          inet addr:192.168.0.133 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::21c:23ff:fe1b:d58f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1256 (1.2 KiB)  TX bytes:1745 (1.7 KiB)
          Interrupt:17

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:32 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1980 (1.9 KiB)  TX bytes:1980 (1.9 KiB)

root@kali:~/Desktop#

```

Note: When running a system from a Live-CD you might have to disable hibernation on your laptop. If the laptop goes into hibernation mode you might experience 'input/output errors' when you try to execute a command. The only solution is to reboot your system.

Basic Linux Shell Commands

man <cmd>	Open the manual page (help) for a command
<cmd> --help	Often invokes simple help instructions for a command
./<cmd>	Execute command in local directory
ls	List directory contents (same as dir)
pwd	Print the current working directory
cd [dir]	Change to a new directory
rm <file>	Remove a file (same as del)
mv <src> <dst>	Move a file
cp <src> <dst>	Copy a file
more <file>	Prints contents of a file to the screen
less <file>	Same as more, but with more options
cat <file>	Copy files (default to <i>stdout</i>)
kate <file> &	Opens file in an advanced text editor (& - background)
ifconfig	Displays network adapter information (IP, MAC, etc.)
dhclient	DHCP client application (e.g., dhclient eth0)
Tab-key	Used to auto-complete commands or directory paths

Note the use of the tab key. This key will accomplish file and command name auto completion. This is a very convenient feature, especially when typing in long filename paths.

Passive Discovery Hands-on Exercise

All the passive discovery hands-on exercises are performed within the Linux command shell.

NOTE: Some of the hands-on exercises require that you replace the string <your network> with your local network address. For example, if the exercise says type “nmap <your network>.1-99” you need to replace <your network> with 1.2.3 if you are on the corporate network, 192.168.10 if you are on the DMZ, and 192.168.0 if you are on the SCADA network.



ARP

One of the simplest ways to identify local network hosts is to review the ARP cache. This can be done by using the arp command:

```
arp -a -i eth0 -n    or    arp -i eth0 -n
```

The results of the command should look similar to Figure 1 where each line shows the IP and MAC addresses for that host.

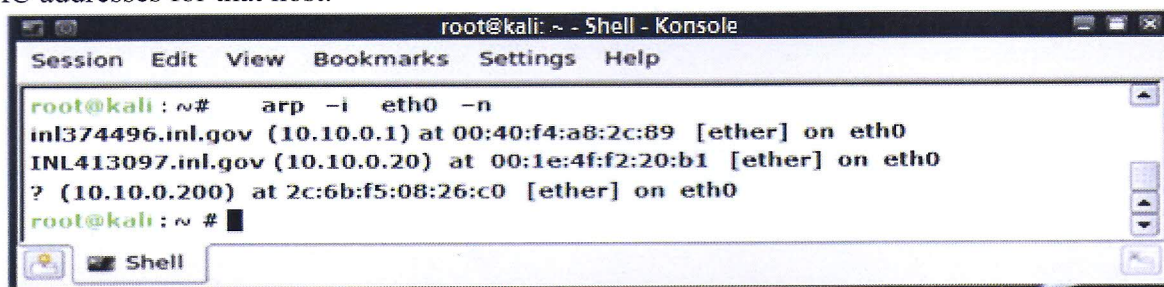


Figure 1 - Screenshot showing the arp cache using the arp -a command.

If the ARP cache is not yet populated, try using the ping command to populate the cache with some local network host data. Ask one or more of your neighbors what their host IP address is.

Next, type the following command where <ip address> is the IP address of your neighbor.

```
ping -c 3 <ip address>
```

The -c indicates the number of ICMP messages to be sent. Now use the up arrow to retrieve your previous arp command, press Enter, and note the additional information in the output.

Netstat

The next reconnaissance method is used to identify local as well as remote hosts and networks by looking at active network sessions using the netstat command. Netstat also shows the processes listening on a given port that helps to identify the network services running on a system. The command options are described as follows: -p owning process ID, -a all sockets, -n no name resolution, -t tcp, -u udp.

```
netstat -pantu
```

The results of this command should look similar to Figure 2, which shows connections for dhclient and rsyslogd over both UDP (IPv4) and UDP6 (IPv6). Try using different netstat command options and see how it affects the output (e.g., netstat -pant, netstat -antu).



Homeland
Security


```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# netstat -pantu  
Active Internet connections (servers and established)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name  
udp        0      0 0.0.0.0:16197           0.0.0.0:*               *          3190/dhclient  
udp        0      0 0.0.0.0:514            0.0.0.0:*               *          2370/rsyslogd  
udp        0      0 0.0.0.0:68             0.0.0.0:*               *          3190/dhclient  
udp6       0      0 :::60797               :::*                    *          3190/dhclient  
udp6       0      0 :::514                 :::*                    *          2370/rsyslogd  
root@kali:~#
```

Figure 2 – Screenshot of current network connections using netstat -pantu command.

You will notice that there is not a lot of information displayed because your host does not have any connections to another host. Go to the icon bar at the bottom of your desktop and select the icon to invoke the FireFox browser. In the browser go to 192.168.10.21 to bring up the ACME HelpDesk page. Run the above netstat command again and notice the established connections that are now listed. The displayed information indicates the IP address, the port, the program, and the program ID of the connection

.bash_history

History and log files may contain information that can be used to identify additional hosts and networks accessible from the current system. The next exercise looks at the bash command history file .bash_history. This file contains a list of all the commands that have been executed in the bash command shell. By identifying commands associated with IP and hostname information we learn about new hosts and networks that could be accessible from this system.

Hence, we are passively discovering and mapping the network as we search through this and other files on the system. The .bash_history file is located in each users' home directory, which is represented by the ~/. To more efficiently search the .bash_history file try using the grep command to search for keywords like ssh, ftp, telnet, etc.

```
grep ssh ~/.bash_history
```

The results of this command should look similar to Figure 3. By using the grep command only the lines that contain the keyword ssh are shown making it easy to identify the hosts likely to be running Linux and have the SSH service running. Try using the same command with a different keyword like telnet, ftp, etc.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# grep ssh ~/.bash_history  
ssh 192.168.10.22  
ssh 10.10.10.123  
ssh 192.168.0.3  
ssh 1.2.3.97
```

Figure 3 – Search for ssh using grep in the .bash_history file.

The `.bash_history` file is not written with recent commands until the terminal (command shell) in which the commands are entered has been exited. Upon exit, the buffered commands are written to the `.bash_history` file. You can test this by exiting your current command shell, starting a new command shell and then typing the command: `cat ~/.bash_history` in the new shell. You will now see all the commands you entered in the command shell before you exited.

Routing Tables

Routing tables are another source for learning about the hosts and networks that are accessible from a system. This information can be used to identify new targets for attack and establish a map of the network. There are four main types of entries in the routing table, host routes, local and remote network routes, and the default route or gateway. To view the routing table use the `route` or `netstat` commands.

```
route -n
```

The results of this command should look similar to the figure below. The Kali Live DVD routing table is not very interesting but you get the idea.

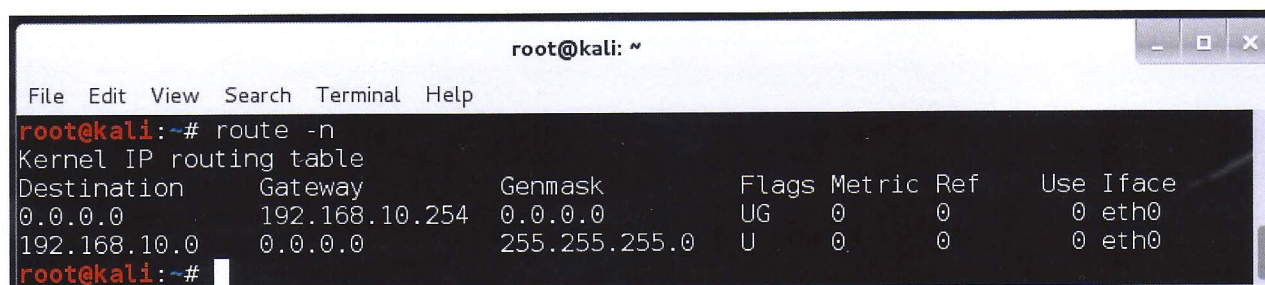


Figure 4 – Routing table shown by using the `route -n` command.

If this system had two network interfaces it may be possible to route traffic between the two networks. This can be determined by looking at the `/proc/sys/net/ipv4/ip_forward` configuration file, with a text viewer or editor, where 1 means it is forwarding and 0 is the inverse. This information is useful for identifying rogue network forwarding or for launching potential man-in-the-middle attacks.

NetBIOS

Another method of reconnaissance can be done with NetBIOS. The Windows `nbtstat -c` command displays the NetBIOS cache, which contains IP addresses. The Kali DVD provides the `nbtscan` tool that can be used to scan a network for NetBIOS information. `Nbtscan` steps outside the realm of passive discovery but it is a tool that utilizes NetBIOS for doing network discovery. Try running this tool by typing the `nbtscan` command with the IP address range to be scanned as shown below.

```
nbtscan <your network>.1-99
```

The results of the command should look similar to Figure 5. Unlike the other exercises to this point, `nbtscan` generates network traffic during discovery.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nbtscan 192.168.10.1-99
Doing NBT name scan for addresses from 192.168.10.1-99
```

IP address	NetBIOS Name	Server	User	MAC address
192.168.10.10	DMZ	<server>	DMZ	00:00:00:00:00:00
192.168.10.41	DOPEY		<unknown>	00:50:56:a0:22:b3
192.168.10.2	SQL	<server>	<unknown>	00:50:56:a0:48:fb
192.168.10.21	JIMMYWIN2KWEBSR	<server>	JIMMYWIN2KWEBSR	00:50:56:a0:2f:e4
192.168.10.32	GUNTHER	<server>	GUNTHER	00:50:56:a0:1c:b6
192.168.10.12	LOCALHOST	<server>	LOCALHOST	00:00:00:00:00:00
192.168.10.66	WIN2KSERVER66	<server>	WIN2KSERVER66	00:50:56:a0:5f:f9
192.168.10.22	JIMMYWINSRV03VM	<server>	<unknown>	00:50:56:a0:1b:d2
192.168.10.50	XSP1	<server>	ADMINISTRATOR	00:50:56:a0:39:b3
192.168.10.97	SPARKY	<server>	<unknown>	00:50:56:a0:2d:b3
192.168.10.42	OARKS	<server>	<unknown>	00:50:56:a0:1d:a2

```
root@kali:~#
```

Figure 5 – NetBIOS network scan using the nbtscan command.

Now that you have issued the nbtscan command, take a look at the arp cache again. Type the command `arp -ni eth0` and note the number of hosts that are now listed.

Post-exercise analysis

- What hosts or networks did you find?

- What ports or services did you find?

More information can be found at:

www.tcpdump.org

More information can be found at:

<http://linux.die.net/man/7/pcap-filter>

or by viewing the man page for "pcap-filter."

Security Note: If you are using tcpdump in a 'unprotected' environment, you must have root/administrator privileges to open the NIC in promiscuous mode, you should use the **-Z <user_name>** option to have tcpdump lower its privilege level to a normal (or less) user account.

FYI – Tcpdump, Packet Capture Library & libpcap were all originally developed at Lawrence Berkeley National Laboratory (LBNL) in Berkeley, California.

More Passive Discovery

Tcpdump/Windump

- Common network traffic capture/analyzer for the command line
- Uses standard libpcap/winpcap to capture/parse network traffic
- Uses Berkeley Packet Filter (BPF) syntax for creating capture filter expressions

A very efficient & clean way for creating a customized "Wire Tap" on your network

Tcpdump's default snap length (the amount of data captured in each packet) is 65,535 bytes. The snap length can be set to limit the number of bytes captured to whatever is desired.

Tcpdump common options

-s <len>	The snap length of the packet capture
-C <size>	Limit output file to size (in MB)
-F <file>	BPF filter file
-i <lan>	Network interface to sniff
-r <file>	Input PCAP file
-w <file>	Output PCAP file
-Z <user_name>	User_name to execute as after opening network interface
-c <num>	Max # of packets to display or write to a file

Berkeley Packet Filter

Tcpdump, and most network sniffing software is built on the Lawrence Berkeley National Laboratories subroutine library known as libpcap. This library provides the capabilities to read network packets from either a network interface or from a previously captured data file often referred to as a pcap file.

A critical part of this library is the Berkeley Packet Filter (BPF) capability. This filtering process allows the user to select which network packets are of interest for further processing by the program.

Don't be intimidated by the syntax. The easiest way to write BPF rules (or any signature rule) is to first think of it as a word sentence, and then convert it to the particular syntax for the application.

A BPF expression consists of one or more primitives. Primitives usually consist of an *id* (name/number) preceded by one or more qualifiers.

There are three types of qualifiers:

- type** – Indicates what the value refers to. Possible types are **host**, **net**, **port**, and **portrange**. If no type is specified, **host** is assumed.
- dir** – Qualifies a particular transfer direction to and/or from the object. Possible directions are **src**, **dst**, **src or dst**, and **src and dst**. For example, “src foo,” “dst net 128.3,” “src or dst port ftp-data.”
- proto** – Restricts the match to a particular protocol. Options are **ether**, **fddi**, **tr**, **wlan**, **ip**, **ip6**, **arp**, **rarp**, **decnet**, **tcp**, and **udp**.

Examples of simple expressions are:

host 10.10.10.1	All traffic involving 10.10.10.1
port 22	All ssh traffic (both UDP and TCP)
src host 10.10.10.1	Traffic originating from 10.10.10.1 (client side ONLY!)
dst host 10.10.10.2	All traffic destined for 10.10.10.2
proto tcp	Only TCP
tcp portrange 1-1024	tcp 'well known' ports

You can build complex expressions using grouping parenthesis, **and**, **or**, and **not**. If you like to use symbolic representation you can also use **&&**, **||**, **!**.

For example:

host foo and not (port 80 or 443)

host foo && ! (port 80 || port 443)

Both of these expressions will filter out all traffic except traffic involving the host foo but it does not involve port 80 or port 443.

Blank lines are ignored. Comments are good things in a filter file and are designated by a **#**. Anything to the right of the **#** is considered a comment. A whole line can be a comment or you can put a comment to the right of a rule.

For example:

Check for active ftp data connection

(not src port 20) # FTP data connection

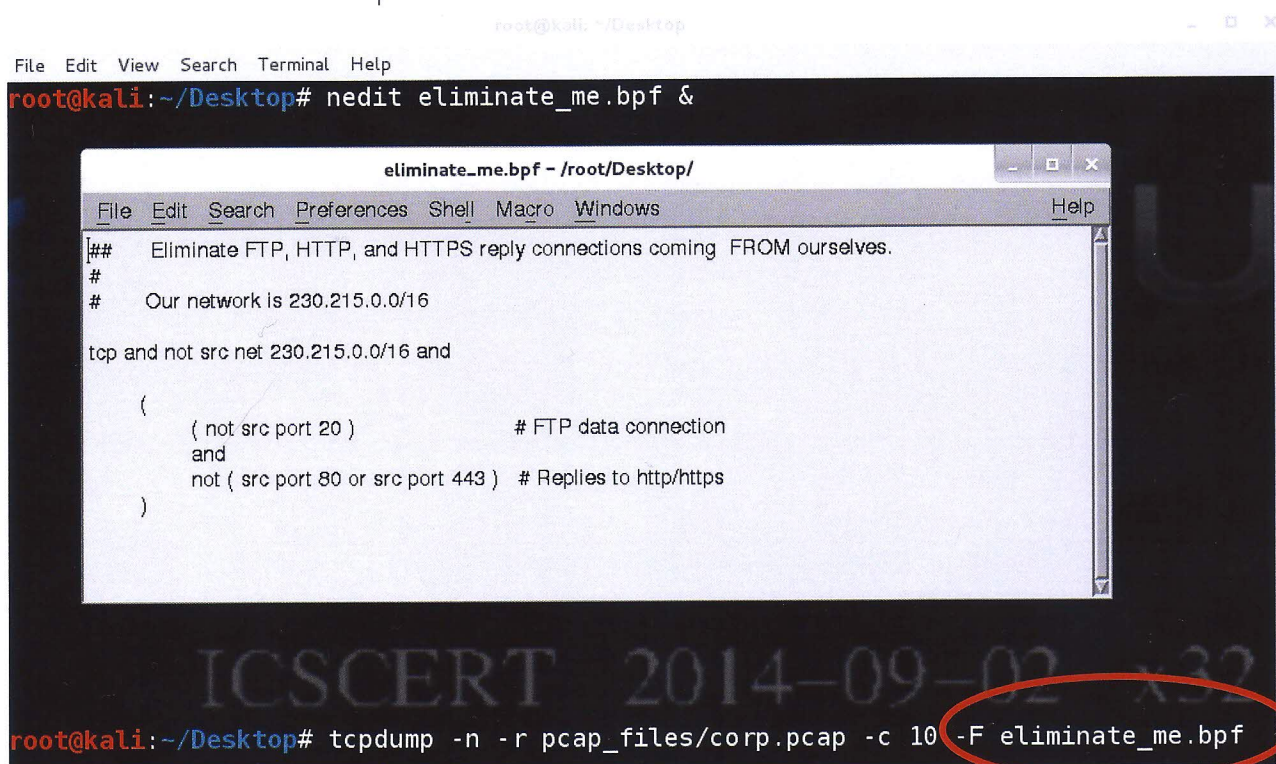
There is also the capability to access the various fields of the packet headers using byte addressing notation in the form of:

proto[starting-byte:lsnumber of bytes]

A BPF can be added to the tcpdump command line at the end of any parameters. To ensure that the UNIX shell doesn't try to mess with it, enclose it in single quotes:

```
tcpdump -ni eth0 (other options) 'host foo && ! ( port 80 || port 443 )'
```

Large BPFs can also be written to a file and then supplied to tcpdump with the -F *filename* option. Below is a sample of a filter file.



The screenshot shows a Kali Linux terminal window with the command `nedit eliminate_me.bpf &` executed. A text editor window titled `eliminate_me.bpf - /root/Desktop/` is open, displaying the following BPF filter rules:

```
## Eliminate FTP, HTTP, and HTTPS reply connections coming FROM ourselves.
#
# Our network is 230.215.0.0/16

top and not src net 230.215.0.0/16 and

(
    ( not src port 20 )          # FTP data connection
    and
    not ( src port 80 or src port 443 ) # Replies to http/https
)
```

Below the editor window, the terminal shows the command `tcpdump -n -r pcap_files/corp.pcap -c 10 -F eliminate_me.bpf`, where the `-F eliminate_me.bpf` part is circled in red. A watermark "ICSCERT 2014-09-02 x32" is visible in the background of the terminal.

Note: The pcap files used in the following examples can be found in the `pcap_files` directory located on the **Desktop** of the Kali DVD.

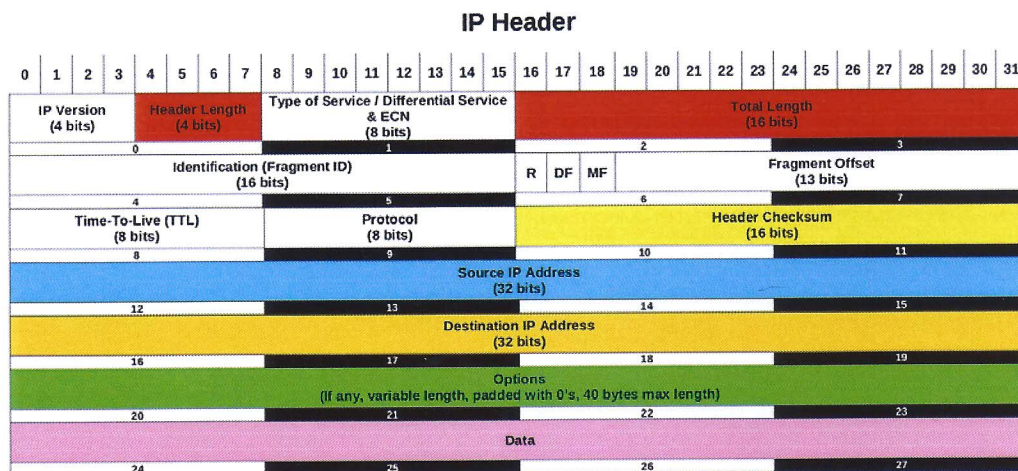
The structure of all TCP/IP headers is based on 32-bit words. Sixty-four bit words were only a dream when the Internet was developed.

For your reference, the *IP*, *TCP*, *UDP*, and *ICMP* header diagrams can be found in the Documents directory.

When referencing parts of any of the headers, always start counting at 0 (zero). To reference consecutive bytes use [S:L], where *S* is the starting byte and *L* is the length. For example, the src ip address in the IP header starts in the 13th byte (ADDRESSED as 12) with a length of 4. Its reference is `ip[12:4]`.

Refer to

`/root/Desktop/Documents/manuals,tutorials,etc/Jstebelon_BPF.pdf` for an extended explanation of addressing in BPF filters.



We can write a BPF filter to check for the LAND attack where the **SRC IP** is equal to the **DST IP** as:

```
ip and ( ip[12:4] = ip[16:4] ) # source ip = destination ip
```

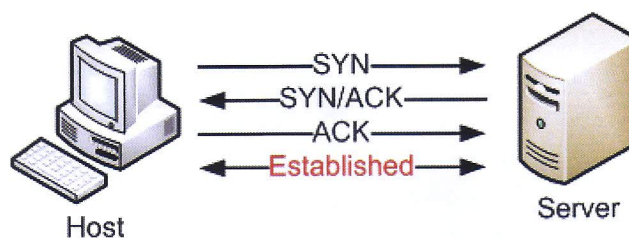
Why would I want to use this filter rule? There is a very famous Denial of Service (DOS) attack that surfaced in 1997 called the **LAND** (Local Area Network Denial) attack. This attack consisted of sending spoofed SYN packets where the victim IP was both the src and dst ip. The result was a locked up system talking to itself. This sounds like ancient history, but the security flaw resurfaced in Windows Server 2003 and Windows XP SP2. This is also a reason why one of the first firewall rules in use today is to check if there is incoming traffic that 'originated' from my own address space.

A quick discussion of TCP will help you understand the next section. Transmission Control Protocol (TCP) is a connection oriented protocol. Think of a phone call. The process for establishing a connection (start of a session) is referred to as the TCP three-way-handshake, as illustrated below.

The steps of the 3-way handshake **are**:

- 1.a Host sends a TCP **SYN**chronize packet to Server.
- 1.b Server receives Host's **SYN**
- 2.a Server sends a **SYN**chronize-**ACK**nowledgement.
- 2.b Host receives Server's **SYN-ACK**.
- 3.a Host sends **ACK**nowledge.
- 3.b Server receives **ACK**.

TCP Three-Step Handshake



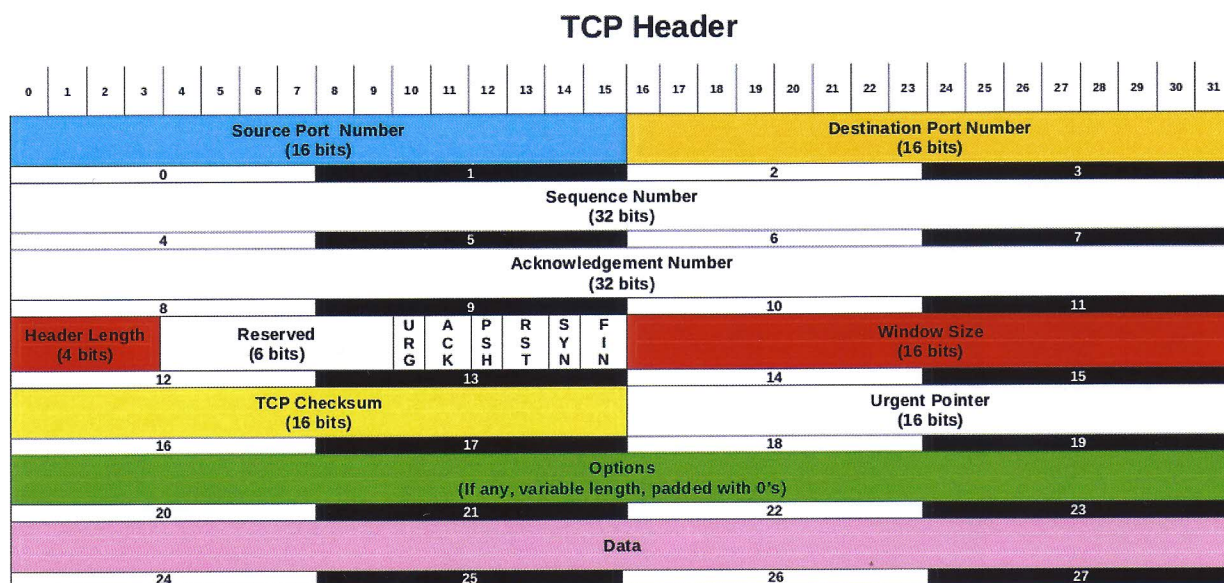
The TCP connection is now established. A similar handshake process is used to end the connection using the **FIN**ish flag instead of the **SYN**chronize flag. The logical connection points are referred to as port. The source port on the HOST is generally assigned by the operating system. The destination port on the server is determined by what service the HOST wants to connect to, eg. ssh, http, https, ftp, etc.

During the conversation the packets that contain data (payload) are flagged using the Push flag.

The flag bits in the TCP header (byte 13) are:

- F: FIN (Ending of sending by sender – Connection termination)
- S: SYN (Synchronize sequence numbers – Connection establishment)
- R: RST (Reset connection)
- P: PSH (Push data- data should be transferred immediately)
- A:ACK (Acknowledgement)
- U:Urgent (High priority scheduling)

Now we can take a quick look at the structure of the TCP header using the figure below. Note, this figure can also be found in the Kali Documents folder, */root/Documents/TCP_header.pdf*.



The **SRC Port Number** starts in **Word 0** of the TCP header and is 2 bytes long (16 bits) bytes 0-1. Looking at the diagram, the TCP flags field (**URG**, **ACK**, **PSH**, **RST**, and **FIN**) are contained in byte 12, bits 7-6 and continues into byte 13, bits 3-0. This is important information to remember since it is often used in the Berkley Packet Filters to allow examination of particular bytes and/or bits of the packet headers. For example the **SYN** flag is found in byte 13 which is expressed as **tcp[13]**.

Another important concept is **Byte Masking** and **Boolean Algebra**. A Byte Mask is constructed by setting the bits you are interested in to 1. As an example, if I want to check to see if the **FIN** bit is set, my byte mask would be 1 (only bit 0 set). So, the **FIN** flag would be **TCP[13] & 1**. (Refer to */root/Desktop/Documents/manuals,tutorials,etc/Jstebelton_BPF.pdf* for an extended explanation of Boolean Algebra in BPF filters.)

BPF Examples

Here are some examples of using BPF filters. Note that some of the output has been truncated to make it easier to read.

The examples reference pcap files that are available in the **pcap_files** directory. The pcap files used during these calls are found in the **pcap_files** on the Desktop. For the example, the “-r pcap_files/scada.pcap” switch tells tcpdump to read the data from the **scada.pcap** file found in the **pcap_files** directory relative to the Desktop.

EXAMPLE 1: Find all DNP3 (port 20000) traffic

tcpdump -n -r pcap_files/scada.pcap -c 2 port 20000

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)
07:34:40.881503 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.],
07:34:40.881869 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [.]

EXAMPLE 2 Find all traffic (conversations) between 117.173.125.61 and 117.173.125.217

tcpdump -n -r pcap_files/scada.pcap -c 2 host 117.173.125.61 and host 117.173.125.217

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)
07:34:40.881503 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.],
07:34:40.881869 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [.]

EXAMPLE 3 Find all conversations that are NOT from 117.173.125.62 but to 117.173.125.217

tcpdump -n -r pcap_files/scada.pcap -c 2 not host 117.173.125.62 and 117.173.125.217

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)
07:34:40.881503 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.],
07:34:40.881869 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [.]

EXAMPLE 4 Find all packets with SYN flag ONLY

tcpdump -n -r pcap_files/scada.pcap -c 2 'tcp[13] = 2'

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)
07:34:39.393199 IP 117.173.125.125.3341 > 117.173.125.61.2081: Flags [S],
07:34:40.221530 IP 117.173.125.125.3342 > 117.173.125.61.2081: Flags [S],

EXAMPLE 5 Match all packets with SYN-ACK

tcpdump -n -r pcap_files/scada.pcap -c 2 'tcp[13] = 18'

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)
07:34:43.128360 IP 255.26.126.235.80 > 181.253.75.153.51134: Flags [S.],
07:34:46.372012 IP 117.173.125.61.2000 > 181.253.75.55.1086: Flags [S.],

Depending on your requirements you can either check the actual value of TCP byte 13 or you can use a mask function to check if a particular bit is set in the byte. For example, to check if the SYN bit is set, ignoring any other bit, we can use `tcp[13] & 2 = 2`. The `& 2` is a bit mask.

Notice in the example above, there isn't any packets found between 117.173.125.61 and 117.173.125.217. **Why not?**

The reason there were no SYN packets found between the two hosts is that DNP3 is a continual conversation. Remember, TCP/IP is modeled after a telephone conversation. You say hello at the beginning and Good-Bye at the end. DNP3 starts with a SYN packet when the systems involved first communicate. It ends when the process is terminated...may be months later!

EXAMPLE 6 All packets between src host 117.173.125.61 and dst host 117.173.125.217
tcpdump -n -r pcap_files/scada.pcap -c 5 src host 117.173.125.61 and dst host 117.173.125.217

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)

```
07:34:40.881503 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq
1644131194:1644131218, ack 3863577870, win 64485, length 24
07:34:41.012371 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [.], ack 18, win 64468, length 0
07:34:41.082696 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq 24:39, ack 18, win
64468, length 15
07:34:42.889154 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq 39:66, ack 18, win
64468, length 27
07:34:43.023958 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [.], ack 106, win 64380, length 0
```

EXAMPLE 7 All packets with src or dst port = 20000 (DNP3)
tcpdump -n -r pcap_files/scada.pcap -c 5 port 20000

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)

```
07:34:40.881503 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq
1644131194:1644131218, ack 3863577870, win 64485, length 24
07:34:40.881869 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [.], ack 24, win 5840, length 0
07:34:40.883297 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [P.], seq 1:18, ack 24, win 5840,
length 17
07:34:41.012371 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [.], ack 18, win 64468, length 0
07:34:41.082696 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq 24:39, ack 18, win
64468, length 15
```

EXAMPLE 8 Create a file with a bpf to be used with tcpdump.
cat << EOF > my.bpf
(host 117.173.125.61 and dst host 117.173.125.217) and port 20000
EOF
tcpdump -n -r pcap_files/scada.pcap -c 5 -F my.bpf

reading from file pcap_files/scada.pcap, link-type EN10MB (Ethernet)

```
07:34:40.881503 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq
1644131194:1644131218, ack 3863577870, win 64485, length 24
07:34:40.881869 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [.], ack 24, win 5840, length 0
07:34:40.883297 IP 117.173.125.217.20000 > 117.173.125.61.2753: Flags [P.], seq 1:18, ack 24, win 5840,
length 17
07:34:41.012371 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [.], ack 18, win 64468, length 0
07:34:41.082696 IP 117.173.125.61.2753 > 117.173.125.217.20000: Flags [P.], seq 24:39, ack 18, win
64468, length 15
```

The table lists the various BPF syntax primitives.

Capture Filter Primitives

<code>[src dst] host <host></code>	Matches a host as the IP source, destination, or either
<code>ether [src dst] host <ehost></code>	Matches a host as the Ethernet source, destination, or either
<code>gateway host <host></code>	Matches packets which used host as a gateway
<code>[src dst] net <network>/<len></code>	Matches packets to or from an endpoint residing in network
<code>[tcp udp] [src dst] port <port></code>	Matches TCP or UDP packets sent to/from port
<code>[tcp udp] [src dst] portrange <p1>--<p2></code>	Matches TCP or UDP packets to/from a port in the given range
<code>less <length></code>	Matches packets less than or equal to length
<code>greater <length></code>	Matches packets greater than or equal to length
<code>(ether ip ip6) proto <protocol></code>	Matches an Ethernet, IPv4, or IPv6 protocol
<code>(ether ip) broadcast</code>	Matches Ethernet or IPv4 broadcasts
<code>(ether ip ip6) multicast</code>	Matches Ethernet, IPv4, or IPv6 multicasts
<code>type (mgt ctl data) [subtype <subtype>]</code>	Matches 802.11 frames based on type and optional subtype
<code>vlan [<vlan>]</code>	Matches 802.1Q frames, optionally with a VLAN ID of vlan
<code>mpls [<label>]</code>	Matches MPLS packets, optionally with a label of label
<code><expr> <relop> <expr></code>	Matches packets by an arbitrary expression

Protocols			Modifiers	Examples	
arp	ip6	slip	! or not	udp dst port not 53	UDP not bound for port 53
ether	link	tcp	&& or and	host 10.0.0.1 && host 10.0.0.2	Traffic between these hosts
fddi	ppp	tr	or or	tcp dst port 80 or 8080	Packets to either TCP port
icmp	radio	udp	ICMP Types		
ip	rarp	wlan	icmp-echo	icmp-routeradvert	icmp-tstampreply
TCP Flags			icmp-unreach	icmp-routersolicit	icmp-ireq
tcp-urg	tcp-rst		icmp-sourcequench	icmp-timxceed	icmp-ireqreply
tcp-ack	tcp-syn		icmp-redirect	icmp-paramprob	icmp-maskreq
tcp-psh	tcp-fin		icmp-echo	icmp-tstamp	icmp-maskreply

Ref: <http://packetlife.net/library/cheat-sheets/>

For more information take a look at <http://biot.com/capstats/bpf.html> or http://www.infosecwriters.com/text_resources/pdf/JStebelton_BPF.pdf

Both of these documents can also be found in the /root/Desktop/Documents Directory, *bpf_syntax.pdf* and *JStebelton_BPF.pdf*, respectively.